MOBILE VISUAL COMPUTING PROJECT

Implementing Discrete Fourier Transform

In this project I have implemented 1d, 2d discrete Fourier transform and also their inverse Fourier transforms, for 1D DFT I have used a periodic function to get the Fourier transform whereas in 2d I have used Images to get the discrete Fourier transform.

Background:

Sir Joseph Fourier was a French scientist who pointed out that periodic function can be expressed as combination of sin and cosine at different frequencies. Later these were also used for finding transforms for non-periodic functions also.

Discrete Fourier Transform:

Discrete Fourier transform in Image processing is used to represent a discrete signal or sequence of data into frequency domain representation.

Implementing Discrete Fourier transform for 1D:

The equation for the Discrete Fourier Transform and for Inverse DFT:

Discrete 1D Fourier Transform

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j\,2\,\pi\,k\,n\,/N}$$

Inverse Discrete Fourier Transform

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, e^{j\,2\,\pi\,k\,n\,/N}$$

Figure 1: The above figure shows the 1-Dimensional DFT formulae's

- I have used the above formula in the implementation of the 1D-DFT and 1D-Inverse-DFT.

Implementing Discrete Fourier transforms for 2D:

The 2D DFT can be written into two 1D-DFT

$$F(u,v) = \sum_{x=0}^{M-1} e^{-j2\pi ux/M} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi vy/N} = \sum_{x=0}^{M-1} F(x,v) e^{-j2\pi ux/M}$$

$$F(x,v) = \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi vy/N}$$

Figure 2: In the above figure we can see that 2D-DFT can be represented by Two 1D-DFT

- By using the above formula, I have computed 2d discrete Fourier transform.

File 1 (Discrete Fourier transform of signal (Random function)):

I have created two files, the first one is 1D-DFT, in that there is a function which is being transformed and the same function is being brought back by using Inverse Fourier transform.

In this function I have taken an line equation and using linspace from NumPy I have simulated it as the signal (with a sample rate) and then computed the DFT for that signal using the above 1-Dimensional DFT formula.

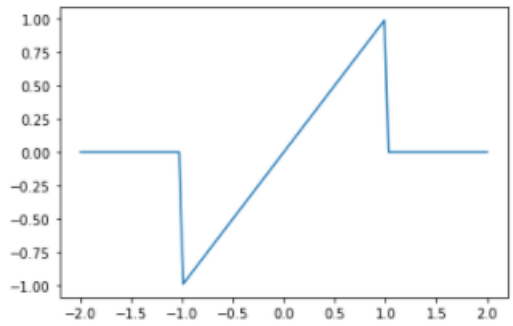

Figure 3: This is the plotted Line

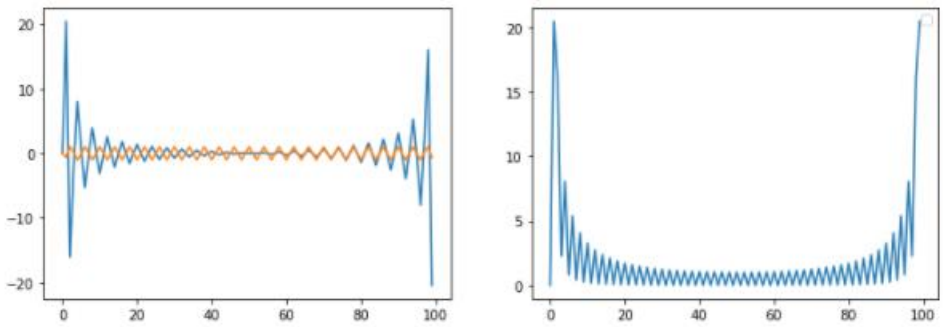

Figure 4: This is the line after DFT with real and Imaginary parts

File 2(Image DFT and Inverse DFT):

In this file I have computed 2d Fourier transform using 50*50 image (which will reduce my computation time) and I have compared the result with NumPy's FFT implementation and the values the image in frequency domain of both NumPy's FFT and my DFT are matching

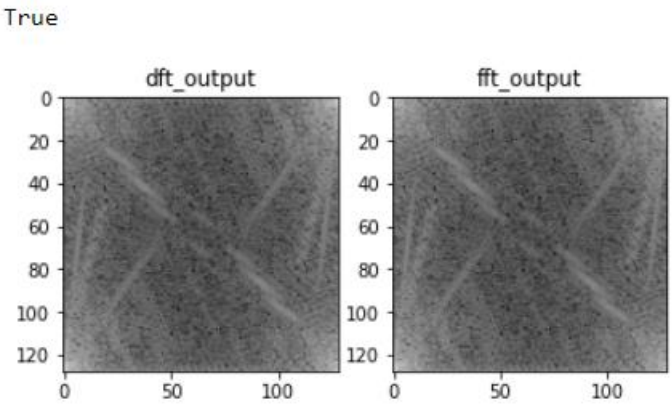

Figure 5: The left side shows my implementation of DFT and right side shows FFT implementation of DFT and the **TRUE** is the similarity in them which was determined by NumPy function.

FILE 3(Masking using FFT(NumPy)):

In this file I have computed Fourier transform of the same image but this time I have implemented Ideal Low filter and Ideal High pass filter on the images and compared the Images and provided the result

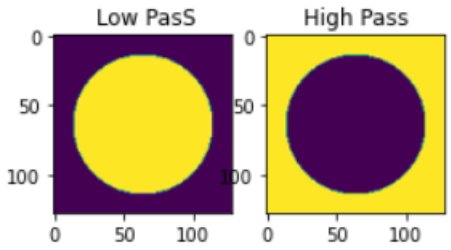

Figure 6: Representation of the Filter

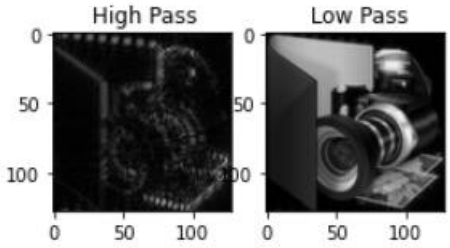

Figure 7: High pass vs Low pass

Acknowledgement:

I express my gratitude towards and I want to thank Professor Yuhao Zhu for his overall support in the Project and learning through subject. I am Indebted for the support and the overall learning I received this Semester.

Conclusion:

Initially I implemented DFT with its standard formula and was using an image with high dimensions and it took very high time for computing the Discrete Fourier Transform, so then I switched to the implementation of Discrete Fourier transform through different Approach of using matrices.